From: garypr7265@worldnet.att.net@inetgw
To: Microsoft ATR
Date: 11/10/01 3:30pm
Subject: Someone thinks you'll like this...

Someone you know, going by the name, "Gary Rost", thought
you'd find this interesting. If you don't like it,
you can yell at them. They said their email address
was garypr7265@worldnet.att.net.

Caught on a planet of peril, he dared challenge its monster rulers... PBS

I, CRINGELY THE PULPIT

NOVEMBER 8, 2001

The Pulpit | I Like It! | Baloney | Old Hat | Tell Me When | Pass It On | Bob's World

Caught in a .NET

Don't Expect Microsoft to Give Up One Weapon Without Acquiring Another -- How .NET Assures the Continuation of Monopoly

By Robert X. Cringely

The proposed Microsoft settlement with the Department of Justice has been out for several days, and there has been more than enough ink used to say that it is a sweetheart deal for Microsoft. The DOJ wants to get on to more important duties like confiscating nail clippers at airports, so the deal looks good to them. But to those of us who got our legal education from old episodes of "Law and Order," the deal stinks. How does it restore competition? What does it do for those hundreds of competitors who are no longer even in business because of Microsoft's monopolistic tactics? Well, those outfits -- if they exist and if they can find the money to do so -- can file civil suits. But most of them won't. I would like to see a class action lawsuit against Microsoft.

What the settlement seems to do is prohibit Microsoft from breaking the law IN THIS SPECIFIC WAY for a period of five years. Imagine a murderer who shot his victims being enjoined for five years from using a gun, but still being allowed to carry a knife. So the best use of this space this week, given all the other pundits who have already criticized the settlement, is for me to throw out some ideas about why Microsoft went for it, and how their behavior will change as a result.

It is important to understand here that Microsoft management does not feel the slightest bit of guilt. They are, as they have explained over and over again, just trying to survive in a brutally competitive industry, one in which they could go from winner to loser in a heartbeat. The fact that Microsoft makes in excess of 90 percent of the profit of the entire software industry, well that's just the happy result of a lot of hard work. Pay no mind to that $36 billion they have in the bank. And since Microsoft doesn't feel guilty, their motivation in agreeing to this settlement is just to get on with business. This is a very important fact to keep in mind when trying to understand the event. This isn't Microsoft being caught and punished, it is Microsoft finding a path back to business as usual, which is to say back to the very kind of practices that got them here. Microsoft, confident in its innate cleverness, is willing to give up certain old monopolistic behaviors because there are new monopolistic behaviors now available to replace them.

Microsoft has to open-up certain Windows communication APIs to other developers, but there is no restriction at all on the addition of new APIs. So expect a LOT of new APIs, many of which will do nothing at all except confuse competitors. There is nothing in the agreement that says Microsoft has to tell anyone which APIs it really intends to use. So just like interpreted software is obfuscated to hinder would-be copiers, expect Microsoft to obfuscate Windows, itself.

Microsoft has to allow third-party middleware, but a glaring loophole was left for Microsoft, simply to redefine code as not being middleware. If they stop distributing code separately and draw it into Windows, well as I read the proposed settlement, middleware stops being middleware after 12 months. So if something new comes up (all the old middleware is explicitly defined) Microsoft can integrate it and screw the opposition one year after they stop distributing it separately.

These loopholes are nice, but they don't amount to the kind of leverage Microsoft would want to have before signing away any rights. Bill Gates would want to believe that he has a new and completely unfettered weapon so powerful that it makes some of the older weapons completely unnecessary. He has found that weapon in .NET.

But hey, .NET isn't even successful yet, right? It might be a big flop. Wrong. Those who think there is any way that .NET won't be universally deployed are ignoring Microsoft's 90 percent operating system market share. Whether people like .NET or not, they'll get it as old computers are replaced with new ones. Within three years .NET will be everywhere whether customers actually use it or not. And that ubiquity, rather than commercial success, is what is important to Microsoft.

Here is the deal. .NET is essentially a giant system for tracking user behavior and, as such, will become Microsoft's most valuable tactical tool. It is a system for tracking use of services, and the data from that tracking is available only to Microsoft.

.NET is an integral part of Windows' communication system with all calls going through it. This will allow Microsoft (and only Microsoft) to track the most frequently placed calls. If the calls are going to a third-party software package, Microsoft will know about it. This information is crucial. With it, Microsoft can know which third-party products to ignore and which to destroy. With this information, Microsoft can develop its own add-in packages and integrate them into the .NET framework, thus eliminating the third-party provider. A year later, as explained above, the problem is solved.

Alternately, Microsoft could use the information (this .NET-generated market research that Microsoft gets for free and nobody else gets at all) to change Windows to do service discovery giving an automatic priority to Microsoft's middleware. The advantage here is in giving the appearance of openness without actually being open.

These possible behaviors are not in any way proscribed by the proposed settlement with the DOJ, yet they virtually guarantee a continuation of Microsoft's monopoly on applications and services as long as Microsoft has an operating system monopoly. When Microsoft talks about "innovation," this is what they mean. Nothing is going to change.

It is easy to criticize, but for a change, there is actually something that you and I can do about this problem. Under the Tunney Act, the court has to open a 60-day period for public comment before any settlement can become final. This will happen after the settlement is entered in the Federal Register and will probably involve the court establishing a web site. This will be your chance to say what you think should happen in the case (look in the "I Like It" links for further information). My preferred outcome is still that Microsoft be forced to sell its language business, and the proceeds of that sale be distributed to registered users of Microsoft products. You might think to suggest that in your comments to the court.

Finally, I have a little space left over to respond to some of the critics of last week's column about the predicted rise of Microsoft's C# language at the expense of Java. This column was wildly unpopular in the technical community. Remember that unpopular is not bad for a columnist. UNREAD is bad. So I thank all those people who got upset and told their friends to read what idiotic things I had written. But I also stand by my words. So here are the typical complaints followed by my typical responses. Thanks go to those nerds who provided such pithy criticism and especially to those who helped me sound halfway intelligent in my response. For those whose eyes glaze over when the talk gets technical, just reread the first part of this column and get mad at Microsoft all over again.

Criticism 1: C# Apps are tied to Windows, Windows, Windows. While this is fine and wonderful for windows developers, there are thousands of UNIX/Mac/mainframe/PalmOS/etc developers out there that will be left high and dry. And let's not forget Java runs on everything from mainframes to smart cards.

Bob's reply: Java won't die and I never said it would. Java didn't kill C++ -- it just stole market share. Visual BASIC was one of the biggest languages of the 90s yet it was Windows-only. Unix is entrenched on the server side, but that's the fault of Windows, not C#. So maybe we can rephrase it -- C# will dominate Java on Windows. That's still a pretty big statement. Not to mention, C# is compiled to an IR, making a C# runtime for Unix possible and even probable as long as it skips serious Windows-specific APIs.

Criticism 2: The "Java is slow" myth. More recent JVMs can actually perform as well as or BETTER than natively compiled code. This is because they do just-in-time compilation, making the Java code as fast as native machine code. Also, there is only so much optimization the compiler can do when you compile a program, having no idea how it will actually be used when it is run. At runtime, there is a lot more information available to the system as to what parts of the code are the real bottlenecks. Recent Java implementations employ dynamic runtime optimizations, where parts of the program that run more frequently are recompiled in an optimized manner to improve performance. These dynamic optimization schemes are a very exciting new field for compiler and virtual machine engineers -- and they are totally lacking from poor old statically compiled C#. The very way that C# gets compiled ties you to Windows, so dynamic optimization of running C# code will be all but impossible to implement. In the long run, Java has the potential to seriously outperform all statically compiled languages.

Bob's reply: I have very informed friends who have ripped JVMs inside out and they can't even see HOW JVMs can be claimed to perform better than compiled code. C (we're not talking about C# yet, because the critic is talking natively compiled code, like C) will always outperform Java. C doesn't have garbage collection, runtime type checking, runtime array bounds checking, dynamic linking, runtime dynamic-optimization, etc. Java just does too much for the user at runtime to ever run as fast as C.

Here's some code:

```
for (int g=0;g<x;++g) a[g] = 10;
```

This code is legal in both C and Java. Java will array check, C will not. Every other conceivable optimization could be applied to either language.

Whether C# will beat Java (or more precisely whether C#'s compiled version beat Java's compiled "bytecode" remains to be seen. But we can expect compilers to compile Java to C#'s compiled representation soon -- it is not a hard problem.

C# does introduce many of the same things that slow down Java. However, Java's fixation on platform independence does cost it in terms of abstraction. That is, it is abstracted farther from the architecture than C# will be (because of C#'s dedication to Windows). We can't say for sure yet, but the C# runtime makers have the experience of the Java runtime makers to build on and can rely what they're underlying architecture is -- Windows. So I'd bet C# will be faster, but for now that's just a bet.

As for just-in-time compilation, it has never, ever, made things as fast as native machine code. Look at Tower Technology's Tower JVM that specializes in compiling Java TO native machine code. They beat JITs easily.

About pre-runtime optimization, it simply doesn't work as elegantly as people would like to believe. Pre-runtime optimizers can spend all week optimizing EVERY spot, not just hotspots. The only time dynamic optimizations come into play is in inlining virtual methods that appear to be non-virtual. DashO-Pro is a static pre-runtime Java optimizer that can inline virtual methods with great success. Hotspot goes a little farther. But C# will also have dynamic optimization. Saying that doing so is impossible is just naïve.

C# will JIT'd in Windows. The dynamic optimizations are sure to be implemented. Being coupled to Windows is irrelevant. In fact, C# will have MORE optimization opportunity because it is tied to Windows and Intel. Heck, a C# runtime could get to the level of avoiding CPU pipeline stalls because it can be sure of what its running on. Java could never come close to that.

When Quake 4 comes out in Java, let me know.

Criticism 3: Java is open. Sun develops Java APIs and technologies in conjunction with hundreds of other companies and individuals around the world. Anyone in the world can implement most Java APIs without paying Sun a dime (now if you want that little coffee logo on your product, that's a different story, the make you pay for interoperability testing for that). While Microsoft seems willing to "standardize" C#, they will probably open up the language while holding the runtime libraries close to the vest. What good would C have been if the standard C runtime libraries were vendor-specific? What this means for developers is a single-vendor solution, just like Windows. A large part of Java's success comes from the fact that you can put together applications by mixing and matching pieces from multiple vendors and be guaranteed easy interoperability. For example, you can build an e-commerce web site by buying a Servlet engine from Allaire, an EJB app server from BEA, and Java database drivers from Oracle -- and they will all work FINE together -- AND you can pick any kind of hardware and operating system! Want your developers to work in Windows, but deploy the app on UNIX? No problem. Want to upgrade from your Intel-based Dell servers to Sun's new 64-CPU UltraSPARC machine? Your code requires NO changes! You don't even need to recompile it, because Java is not statically compiled! What's Microsoft's answer to this? Run everything Microsoft: ASP, IIS, ADO, etc. Develop the app on Windows. Deploy the app on Windows. Stay with Windows forever, and hope Microsoft is good about fixing the plethora of bugs and security holes that will inevitably arise. With C#, who will supply the runtime libraries?

Bob's reply: Java's strength surely is its standardized APIs. C might have that for standard libraries, but branch off into networking or HTTPS and you have plenty to choose from, which is the problem. C# will have standard libraries too. Just as Sun provided all of Java's libs (users had no say) MS will provide all of C#'s.

The idea that software components from different vendors can be mixed together and work FINE is an oversimplification. Even getting Java applets to work in Netscape and IE at the same time can be a major pain. Java's platform independence is dubious. It seems like 95 percent of your application will work perfectly when moved to another platform like Unix and it takes another week to fix five percent of the quirks. No problem? I disagree.

Java is "statically compiled", but it is not statically linked. And even if it was statically linked, it runs everywhere because it is compiled in a generic stack-based intermediate representation called bytecode. Bytecode is then interpreted or JIT'd by some Java runtime build specifically for a given architecture. Java runtimes cannot be built in Java, they are usually written in C or C++.

Microsoft already has supplied the runtime libraries for C#. Check the docs for .NET. Also, we're sort of forgetting here that C# compiles to CIL (an intermediate representation) but so does Visual BASIC and C++. In other words, modules between these languages will interoperate seamlessly. Where Java gives you platform independence, .NET is giving you language independence. I fully expect Java to be added to the interoperability list too.

It is too early to be sure, but is highly likely that existing C++ and VB code can be recompiled under Visual Studio 7 and then called and used by C# programs. That is, every library written in VB or C++ is already a .NET library.

Criticism 4: Developers have learned long ago that single-vendor lock-in solutions are a recipe for disaster. If you can't swap out a buggy piece with a functionally correct one from a different vendor, you're tied to the poor-quality vendor (like Microsoft).

Bob's reply: And 90 perecent of the world runs Windows because...?

OS/2 was better than Windows, but that didn't matter. This is about marketing and company strength. Microsoft has $36 billion in the bank while Sun is laying people off.

Criticism 5: Do not discount Java simply because you don't see lots of consumer applications written in Java. Java has serious momentum on the server side.

Bob's reply: Java GUI is dead. Go to Best Buy and look for Java apps. They aren't there. Java's platform independence was all about satisfying a million clients running Windows, Unix, Mac, etc. and then we abandoned Java on the client because it was too slow and klunky.

Now we tout Java platform independence on the server as if we never had the foresight to buy the right server in the beginning. Or even if we must upgrade, we must upgrade to a radically different architecture. Sure, platform independence on the server is great, but is nowhere near as grand as the original vision of PI on the client.

Java's true strength is its programmer productivity. It's broad set of standardized libraries make programmers far more productive than C++ or C. Unfortunately, C# can match that. Technical merits and zealotry aside -- Sun would BE Microsoft if they could. Even if Java had wonderful technical merit over C# and .NET (which it really doesn't), it would lose. That's business.

Home The Pulpit I Like It Baloney Old Hat Tell Me When Pass It On Bob's World